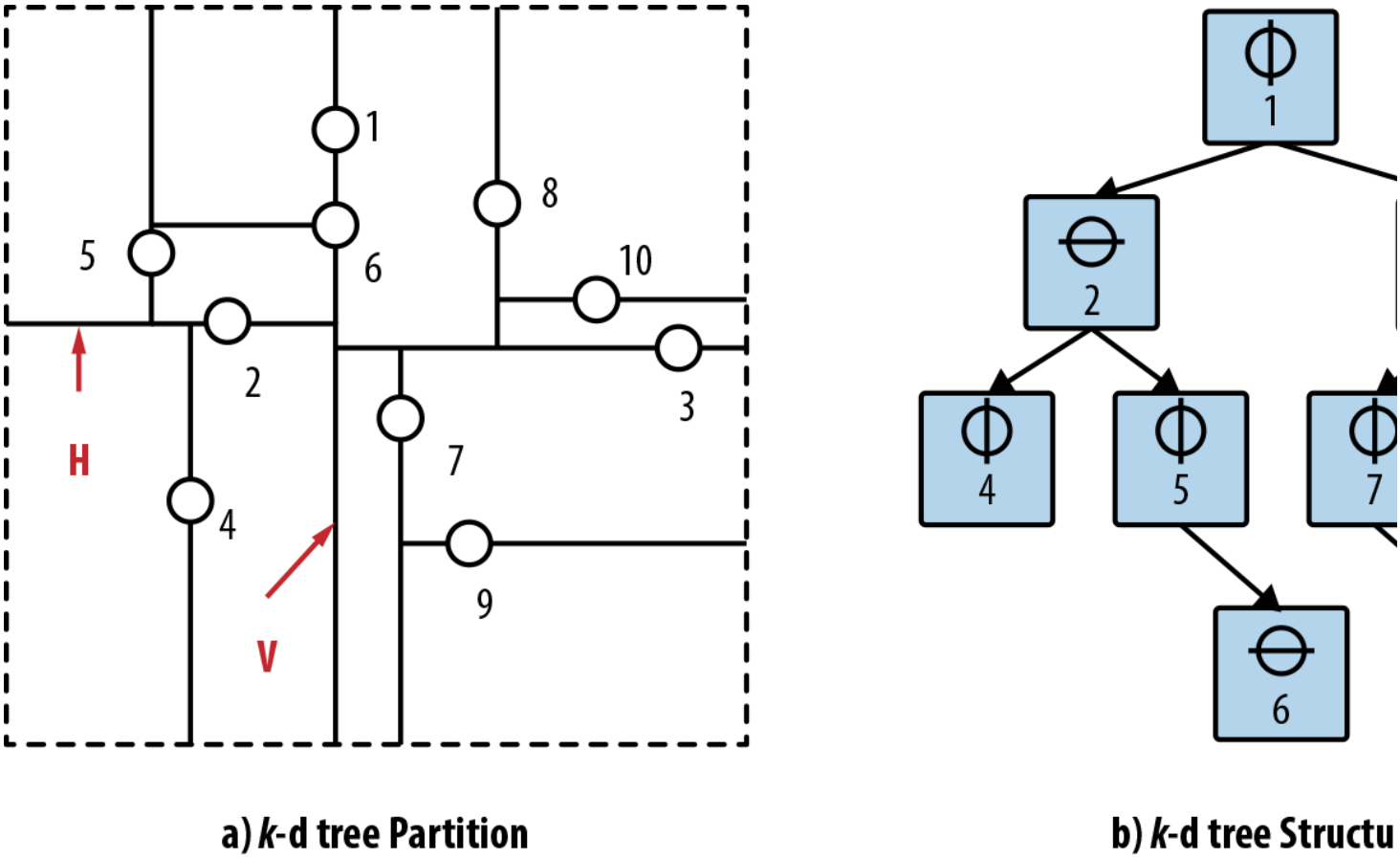
K-d tree project description

Introduction

A 2-dimensional k-d tree is a spatial tree structure that shows how to represent data efficiently to support the execution of nearest neighbor queries. A k-d tree subdivides a k-dimensional plane along the perpendicular axes of the coordinate system. These points are numbered in the order in which they were inserted into the tree. The structure of the k-d tree below is depicted as a binary tree.



a) *k*-d tree Partition b) *k*-d tree Structu

A k-d tree is a recursive binary tree structure where each node contains a point and a coordinate label (i.e., either x or y) that determines the partitioning orientation. The root node represents the rectangular region (x_{low} = -INFINITY, y_{low} = -INFINITY, x_{high} = +INFINITY, y_{high} = +INFINITY) in the plane partitioned along the vertical line V through point p_1. The left subtree further partitions the region to the left of V, whereas the right subtree further partitions the region to the right of V. The left child of the root represents a partition along the horizontal line H through p_2 that subdivides the region to the left of V into a region above the line H and a region below the line H. The region (-INFINITY, -INFINITY, p_1.x, +INFINITY) is associated with the left child of the root, whereas the region (p_1.x, -INFINITY, +INFINITY, +INFINITY) is associated with the right child of the root. These regions

are effectively nested, and we can see that the region of an ancestor node wholly contains the regions of any of its descendant nodes.

In a nearest neighbor query, the goal is to find the point p in P that is closest to a target point x. The brute force implementation would simply compute the distance to x from all points in P and return the one that is the closest. This performance is clearly O(*n*).

Objectives

The objective of this group is to

1. Locate original publication in the literature that described k-d trees and read it. Be sure to cite it in your final report
2. Implement k-d tree in Java. Feel free to use the resources available on the web. In my 2nd edition of Algorithms in a Nutshell, I have an implementation that supports any number of dimensions and that is unnecessary for you. I would ask you to implement a minimal two-dimensional kd-tree.
3. Implement a nearest neighbor query that returns a point in P that that is closest to the target x. If multiple points are equidistant, then you can return any of them.
4. Identify a worst-case data set for inserting N points into a kd-tree
5. Evaluate performance of your algorithm on data sets of size N from 2^5 (that is, 32) doubling to 2^{18} (that is, 262144)
6. Evaluate performance of brute force algorithm on same data sets
7. For what values of N does the k-d tree query capability outperform the brute force approach on random data sets of size N

Deliverables

The deliverables include

1. A working code repository with all code
2. Written report that summarizes the results you discovered, at the level of detail you might find in one of our homework assignments.
3. A 5-minute video presentation (make a zoom presentation recording?) of what you discovered.

Advice

1. Choose this group only if you feel comfortable with binary tree data structure
2. Choose this group only if you are ready to write new code
3. I would rate the difficulty of this task as High